```
/*
Display.cpp - A simple track GPS to SD card logger. Display module.
TinyTrackGPS v0.14

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "Display.h"

Display::Display(Display_Type t):_screen(t){
    _width = 16;
    _height = (_screen > 0) ? 2 : 8;
}

void Display::start(){
    #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd = new LiquidCrystal(LCD_RS, LCD_ENABLE, LCD_D0, LCD_D1, LCD_D2, LCD_D3);
        lcd->begin(_width, _height);
    #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd = new LiquidCrystal_I2C(I2C,_width,_height);
        lcd->init();
        lcd->backlight();
    #endif

    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        // DEFINICION DE CARACTERES PERSONALIZADOS
        static byte alt[8] = { 0x04, 0x0E, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04 };
        static byte ant[8] = { 0x0E, 0x11, 0x15, 0x11, 0x04, 0x04, 0x0E, 0x00 };
        static byte sd[8] = { 0x0E, 0x11, 0x1F, 0x00, 0x00, 0x17, 0x15, 0x1D };
        static byte hourglass_0[8] = { 0x1F, 0x0E, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
        static byte hourglass_1[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
0x1F };
        static byte hourglass_2[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_3[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
        static byte hourglass_4[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0E, 0x0E,
0x1F };
        lcd->createChar(0, hourglass_0);
        lcd->createChar(1, hourglass_1);
        lcd->createChar(2, hourglass_2);
        lcd->createChar(3, hourglass_3);
```

```
        lcd->createChar(4, hourglass_4);
        lcd->createChar(5, alt);
        lcd->createChar(6, ant);
        lcd->createChar(7, sd);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
        u8x8_SSD1306->begin();
        u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
    #endif

    #if defined(DISPLAY_TYPE_SH1106_128X64)
        u8x8_SH1106 = new U8X8_SH1106_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL, SDA);
        u8x8_SH1106->begin();
        u8x8_SH1106->setFont(u8x8_font_7x14B_1x2_r);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display = new DisplaySSD1306_128x64_I2C(-1);
        display->begin();
        //display->setFixedFont(ssd1306xled_font8x16);
        //display->setFixedFont(ssd1306xled_font6x8);
        display->setFixedFont(TinyTrackGPS_font8x16);
    #endif
    this->clr();
}

void Display::clr(){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->clear();
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->clear();
    #endif

    #if defined(DISPLAY_TYPE_SH1106_128X64)
        u8x8_SH1106->clear();
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->clear();
    #endif
}

void Display::print(int vertical, int horizontal, const char text[]){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(vertical, horizontal);
        this->print(text);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64)
        //u8x8_SSD1306->drawString(vertical, (horizontal*2),text);
        u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        this->print(text);
    #endif

    #if defined(DISPLAY_TYPE_SH1106_128X64)
        u8x8_SH1106->setCursor(vertical, (horizontal*2));
```

```
117         this->print(text);
118     #endif
119 
120     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
121         display->setTextCursor((vertical*8),(horizontal*16));
122         this->print(text);
123         //display->printFixed((vertical*8),(horizontal*16),text);
124     #endif
125 }
126 
127 void Display::print(int line, const char text[]){
128     byte pos = _width -(strlen(text));
129     pos = (pos >> 1);
130     this->print((int)pos, line, text);
131 }
132 
133 void Display::print(const char text[]){
134     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
135         lcd->print(text);
136     #endif
137 
138     #if defined(DISPLAY_TYPE_SDD1306_128X64)
139         u8x8_SSD1306->print(text);
140         u8x8_SSD1306->flush();
141     #endif
142 
143     #if defined(DISPLAY_TYPE_SH1106_128X64)
144         u8x8_SH1106->print(text);
145         u8x8_SH1106->flush();
146     #endif
147 
148     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
149         display->write(text);
150     #endif
151 }
152 
153 void Display::print(const char text1[], const char text2[]){
154     this->print((_screen > 0)?0:1, text1);
155     this->print((_screen > 0)?1:2, text2);
156 }
157 
158 void Display::print(const char text1[], const char text2[], const char text3[]){
159     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
160         this->print(text1, text2);
161         delay(750);
162         this->clr();
163         this->print(0,text3);
164     #endif
165 
166     #if defined(DISPLAY_TYPE_SDD1306_128X64) ||
defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx) || defined(DISPLAY_TYPE_SH1106_128X64)
167         this->print(0, text1);
168         this->print(1, text2);
169         this->print(2, text3);
170     #endif
171 }
172 
173 void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){
174     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
175         this->print(text1,text2);
```

```
        delay(750);
        this->clr();
        this->print(text3,text4);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64) || defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx) || defined(DISPLAY_TYPE_SH1106_128X64)
        this->print(0, text1);
        this->print(1, text2);
        this->print(2, text3);
        this->print(3, text4);
    #endif
}

void Display::wait_anin(unsigned int t){
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(15,1);
        lcd->write((byte)t%5);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64)
        const char p[4] = {(char)47,(char)45,(char)92,(char)124};
        u8x8_SSD1306->setCursor((_width-1),6);
        u8x8_SSD1306->print(p[t%4]);
        /*
        static uint8_t hourglass_UP[5][8] = {0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
                                             0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
                                             0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
                                             0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
                                             0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
                                             };

        static uint8_t hourglass_DOWN[5][8] = {0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
                                             0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
                                             0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
                                             0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
                                             0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
                                             };
        u8x8_SSD1306->drawTile((_width-1), 6, 1, hourglass_UP[t%5]);
        u8x8_SSD1306->drawTile((_width-1), 7, 1, hourglass_DOWN[t%5]);
        */
    #endif

    #if defined(DISPLAY_TYPE_SH1106_128X64)
        const char p[4] = {(char)47,(char)45,(char)92,(char)124};
        u8x8_SH1106->setCursor((_width-1),6);
        u8x8_SH1106->print(p[t%4]);
    #endif

    #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
        display->setTextCursor(0,48);
        display->printChar((char)(t%3)+58);
    #endif
}

void Display::print_PChar(byte c) {
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->write((byte)c);
    #endif
```

```
234
235     #if defined(DISPLAY_TYPE_SDD1306_128X64)
236         static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
237                                           0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
238                                           0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
239                                           };
240         static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
241                                           0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
242                                           0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
243                                           };
244         if (c == 5) {
245             u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
246             u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
247         }
248         else if (c == 6) {
249             u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
250             u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
251         }
252         else if (c == 7) {
253             u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
254             u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
255         }
256     #endif
257
258     #if defined(DISPLAY_TYPE_SH1106_128X64)
259         static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
260                                           0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
261                                           0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
262                                           };
263         static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
264                                           0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
265                                           0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
266                                           };
267         if (c == 5) {
268             u8x8_SH1106->drawTile(9, 2, 1, PChar_UP[0]);
269             u8x8_SH1106->drawTile(9, 3, 1, PChar_DOWN[0]);
270         }
271         else if (c == 6) {
272             u8x8_SH1106->drawTile(11, 0, 1, PChar_UP[1]);
273             u8x8_SH1106->drawTile(11, 1, 1, PChar_DOWN[1]);
274         }
275         else if (c == 7) {
276             u8x8_SH1106->drawTile(15, 0, 1, PChar_UP[2]);
277             u8x8_SH1106->drawTile(15, 1, 1, PChar_DOWN[2]);
278         }
279     #endif
280
281     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
282         display->print((char)(c+86));
283     #endif
284 }
285
286 void Display::DrawLogo() {
287     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
288         //display->drawBitmap1(48,24,32,32,Logo_32x32);
289         //display->drawBitmap1(32,18,96,16,TinyTrackGPS_96x16);
290         this->print(4,0,VERSION);
291         this->print(6,1,"^_`a");
292         this->print(6,2,"bcde");
293     #endif
294 }
```

```
295
296 void Display::drawbattery(uint8_t level){
297     #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
298         uint8_t y = 60 - level;
299         NanoRect batt = { {122, y} , {125, 60} };
300         this->print(14, 2, ",=");
301         this->print(14, 3, "+>");
302         if(level>0) display->fillRect(batt);
303     #endif
304 }
305
306 #if defined(DISPLAY_TYPE_SDD1306_128X64_lcdgfx)
307 const PROGMEM uint8_t TinyTrackGPS_font8x16[] = {
308     0x00, // 0x00 means fixed font type - the only supported by the library
309     0x08, // 0x08 = 8 - font width in pixels
310     0x10, // 0x10 = 16 - font height in pixels
311     0x29, // Start char. (41)
312     // Chars for 'Lati' and 'Long' text in vertical
313     0x00, 0x00, 0x48, 0x60, 0x48, 0x48, 0x6c, 0x00, 0x00, 0x00, 0x40, 0x40, 0x46,
    0x4a, 0x6f, 0x00, // Code for char )
314     0x00, 0x00, 0x00, 0x06, 0xea, 0xae, 0xa2, 0x0c, 0x00, 0x00, 0x40, 0x40, 0x4e,
    0x4a, 0x6f, 0x00, // Code for char *
315     // Chars for 'Charge%' text in vertical.
316     0x00, 0x00, 0x62, 0x14, 0x74, 0x74, 0x00, 0x00, 0x34, 0x44, 0x44, 0x46, 0x45,
    0x35, 0x00, 0x00, // Code for char +
317     0x00, 0x64, 0x68, 0x10, 0x2c, 0x4c, 0x00, 0x00, 0x00, 0x00, 0x46, 0xee, 0xa8,
    0x6e, 0x20, 0xc0, // Code for char ,
318     // Chars numbers and signs.
319     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01,
    0x01, 0x01, 0x01, // - 45
320     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00,
    0x00, 0x00, 0x00, // . 46
321     0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x04, 0x04, 0x04, 0x04, 0x04,
    0x04, 0x04, 0x00, // '/'->'=' 47
322     0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20,
    0x10, 0x0F, 0x00, // 0 48
323     0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20,
    0x20, 0x00, 0x00, // 1 49
324     0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x30, 0x28, 0x24, 0x22,
    0x21, 0x30, 0x00, // 2 50
325     0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20,
    0x11, 0x0E, 0x00, // 3 51
326     0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24,
    0x3F, 0x24, 0x00, // 4 52
327     0x00, 0xF8, 0x08, 0x88, 0x88, 0x08, 0x08, 0x00, 0x00, 0x19, 0x21, 0x20, 0x20,
    0x11, 0x0E, 0x00, // 5 53
328     0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20,
    0x11, 0x0E, 0x00, // 6 54
329     0x00, 0x38, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00,
    0x00, 0x00, 0x00, // 7 55
330     0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21,
    0x22, 0x1C, 0x00, // 8 56
331     0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x31, 0x22, 0x22,
    0x11, 0x0F, 0x00, // 9 57
332     // Chars for wait animation.
333     0x01, 0x1f, 0x7f, 0xff, 0xff, 0x7f, 0x1f, 0x01, 0x80, 0xf8, 0x86, 0x81, 0x81,
    0x86, 0xf8, 0x80, // ':'->wait1 58
334     0x01, 0x1f, 0x79, 0xf1, 0xf1, 0x79, 0x1f, 0x01, 0x80, 0xf8, 0xe6, 0xf1, 0xf1,
    0xe6, 0xf8, 0x80, // ';'->wait2 59
335     0x01, 0x1f, 0x61, 0x81, 0x81, 0x61, 0x1f, 0x01, 0x80, 0xf8, 0xfe, 0xff, 0xff,
    0xfe, 0xf8, 0x80, // '<'->wait3 60
```

```
    // Chars for battery icon.
    0xfc, 0x02, 0x03, 0x03, 0x03, 0x03, 0x02, 0xfc, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, // Code for char =
    0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0x7f, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x7f, // Code for char >
    // Chars for display space (' ') and 'm' char.
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ? ->' ' 63
    0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x3F, 0x20, 0x00, 0x3F, // @ ->'m' 64
    // Letters chars for UTM Zone.
    0x00, 0x00, 0xC0, 0x38, 0xE0, 0x00, 0x00, 0x00, 0x20, 0x3C, 0x23, 0x02, 0x02, 0x27, 0x38, 0x20, // A 33
    0x08, 0xF8, 0x88, 0x88, 0x88, 0x70, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x11, 0x0E, 0x00, // B 34
    0xC0, 0x30, 0x08, 0x08, 0x08, 0x08, 0x38, 0x00, 0x07, 0x18, 0x20, 0x20, 0x20, 0x10, 0x08, 0x00, // C 35
    0x08, 0xF8, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x10, 0x0F, 0x00, // D 36
    0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x23, 0x20, 0x18, 0x00, // E 37
    0x08, 0xF8, 0x88, 0x88, 0xE8, 0x08, 0x10, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03, 0x00, 0x00, 0x00, // F 38
    0xC0, 0x30, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x07, 0x18, 0x20, 0x20, 0x22, 0x1E, 0x02, 0x00, // G 39
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x21, 0x01, 0x01, 0x21, 0x3F, 0x20, // H 40
    0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, // I 41
    0x00, 0x00, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x00, 0xC0, 0x80, 0x80, 0x80, 0x7F, 0x00, 0x00, 0x00, // J 42
    0x08, 0xF8, 0x88, 0xC0, 0x28, 0x18, 0x08, 0x00, 0x20, 0x3F, 0x20, 0x01, 0x26, 0x38, 0x20, 0x00, // K 43
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x20, 0x30, 0x00, // L 44
    0x08, 0xF8, 0xF8, 0x00, 0xF8, 0xF8, 0x08, 0x00, 0x20, 0x3F, 0x00, 0x3F, 0x00, 0x3F, 0x20, 0x00, // M 45
    0x08, 0xF8, 0x30, 0xC0, 0x00, 0x08, 0xF8, 0x08, 0x20, 0x3F, 0x20, 0x00, 0x07, 0x18, 0x3F, 0x00, // N 46
    0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x20, 0x10, 0x0F, 0x00, // O 47
    0x08, 0xF8, 0x08, 0x08, 0x08, 0x08, 0xF0, 0x00, 0x20, 0x3F, 0x21, 0x01, 0x01, 0x01, 0x00, 0x00, // P 48
    0xE0, 0x10, 0x08, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x0F, 0x18, 0x24, 0x24, 0x38, 0x50, 0x4F, 0x00, // Q 49
    0x08, 0xF8, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x03, 0x0C, 0x30, 0x20, // R 50
    0x00, 0x70, 0x88, 0x08, 0x08, 0x08, 0x38, 0x00, 0x00, 0x38, 0x20, 0x21, 0x21, 0x22, 0x1C, 0x00, // S 51
    0x18, 0x08, 0x08, 0xF8, 0x08, 0x08, 0x18, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // T 52
    0x08, 0xF8, 0x08, 0x00, 0x00, 0x08, 0xF8, 0x08, 0x00, 0x1F, 0x20, 0x20, 0x20, 0x20, 0x1F, 0x00, // U 53
    0x08, 0x78, 0x88, 0x00, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x07, 0x38, 0x0E, 0x01, 0x00, 0x00, // V 54
    0xF8, 0x08, 0x00, 0xF8, 0x00, 0x08, 0xF8, 0x00, 0x03, 0x3C, 0x07, 0x00, 0x07, 0x3C, 0x03, 0x00, // W 55
    0x08, 0x18, 0x68, 0x80, 0x80, 0x68, 0x18, 0x08, 0x20, 0x30, 0x2C, 0x03, 0x03, 0x2C, 0x30, 0x20, // X 56
    0x08, 0x38, 0xC8, 0x00, 0xC8, 0x38, 0x08, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x00, 0x00, 0x00, // Y 57
```

```
368     0x10, 0x08, 0x08, 0x08, 0xC8, 0x38, 0x08, 0x00, 0x20, 0x38, 0x26, 0x21, 0x20,
    0x20, 0x18, 0x00, // Z 58
369     // Personalized simbols for display info.
370     0x30, 0x38, 0x3c, 0xff, 0xff, 0x3c, 0x38, 0x30, 0x00, 0x00, 0x00, 0xff, 0xff,
    0x00, 0x00, 0x00, // '['->altitud 91
371     0x3c, 0x02, 0x01, 0xd9, 0xd9, 0x01, 0x02, 0x3c, 0x00, 0xc0, 0xe0, 0xff, 0xff,
    0xe0, 0xc0, 0x00, // '\'->antena 92
372     0x78, 0x7c, 0x6e, 0x66, 0x66, 0x6e, 0x7c, 0x78, 0x7c, 0xfc, 0xc0, 0xf8, 0x7c,
    0x0c, 0xfc, 0xf8, // ']'->sd 93
373     // Chars as logo.
374     0x00, 0x00, 0x80, 0xC0, 0x60, 0x10, 0x98, 0x4C, 0xF0, 0x1E, 0x03, 0xF0, 0x0C,
    0x03, 0x81, 0x00, // char 94
375     0x64, 0x26, 0x12, 0x12, 0x0B, 0x09, 0x09, 0x49, 0x00, 0x00, 0x00, 0x00, 0x80,
    0xC0, 0x60, 0xB0, // char 95
376     0x49, 0x09, 0x09, 0x0B, 0x12, 0x12, 0x26, 0x64, 0x10, 0x18, 0x08, 0xC4, 0x64,
    0x1E, 0x07, 0x03, // char 96
377     0x48, 0x98, 0x10, 0x60, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x81, 0x03, 0x0C, 0xF0,
    0x03, 0x1E, 0xF0, // char 97
378     0x0F, 0x78, 0xC0, 0x0F, 0x30, 0xC0, 0x81, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06,
    0x08, 0x19, 0x32, // char 98
379     0xC0, 0xE0, 0x78, 0x26, 0x23, 0x10, 0x18, 0x08, 0x26, 0x64, 0x48, 0x48, 0xD0,
    0x90, 0x90, 0x92, // char 99
380     0x0D, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x92, 0x90, 0x90, 0xD0, 0x48,
    0x48, 0x64, 0x26, // char 100
381     0x00, 0x81, 0xC0, 0x30, 0x0F, 0xC0, 0x78, 0x0F, 0x32, 0x19, 0x08, 0x06, 0x03,
    0x01, 0x00, 0x00, // char 101
382
383 };
384 #endif
```